情報理工学部　SN コース 2 回
セキュリティ・ネットワーク学実験 2
課題 4-4 レポート

2600200443-6
Yamashita Kyohei
山下 恭平

Oct 17 2021

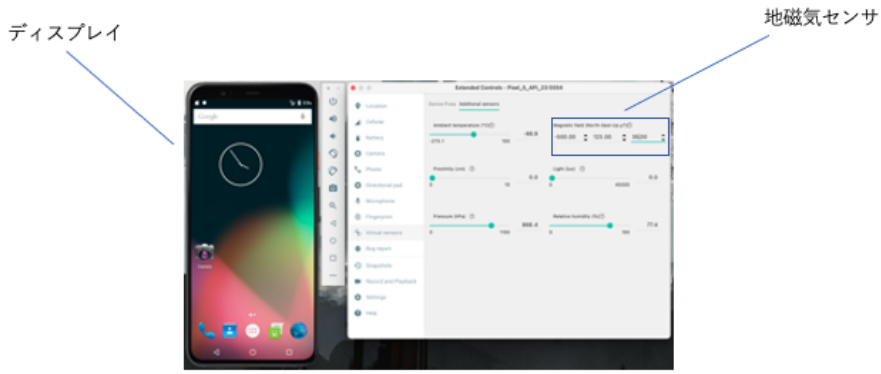
1　概要

Android Studio と Android のエミュレーターを用いて、Android 端末に搭載されているセンサから取得した情報を、端末上に表示するプログラムの開発、および、取得した情報をリアルタイムでグラフとして画面上に表示するアプリケーションの開発。

2　外部仕様

2.1　開発対象の使い方に関する説明

この実験ではアンドロイドスマートフォンをエミュレーターを用いて利用している。
主に使用する機能は、ディスプレイおよび地磁気センサである。地磁気センサの値はエミュレーターの設定により自由に設定することが可能である。



2.2　開発対象を構成するハードウェアと、その主な仕様

今回主に使用したハードウェアは地磁気センサである。
地磁気センサは x 軸、y 軸、z 軸それぞれの磁気を測定し、測定した値を要素数 3 の配列として返す。
ディスプレイはグラフの出力先として使用した。

表 1　ハードウェア一覧

機器一覧	使用/情報
Android スマートフォン	エミュレーターにより実装、ver:Android6.0
地磁気センサ	xyz 軸それぞれの磁気を測定、要素数 3 の配列を返す
ディスプレイ	センサの情報、グラフの出力先

2.3　ハードウェアやソフトウェアが担当する機能と、機能同士の関連

今回使用したハードウェアは地磁気センサとディスプレイの二つである。センサから取得した情報をディスプレイに出力するまでの処理をプログラム (ソフトウェア) で実装している。
プログラムはセンサからの値を取得する部分と、その値をグラフへと出力する部分で構成されている。
以下に機能構成図を示す。

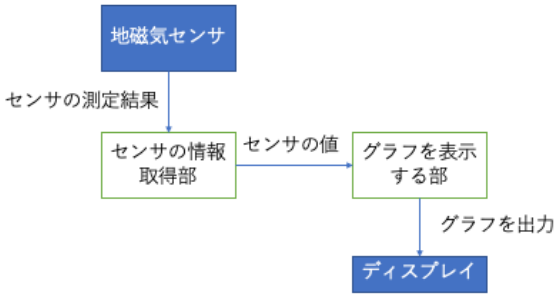


図 1　機能構成図

2.4　開発に用いたプログラミング言語と開発環境

開発に使用したものを以下の表にまとめる.

表 2　開発に使用したもの

OS	macOS Big Sur
開発環境	Android Stusio
使用した言語	Java

3　内部仕様

3.1　各ハードウェアが備えるソフトウェアの詳細な設計

地磁気センサから情報を取得するために、要素数 3、float 型の配列をあらかじめ準備しておく。地磁気センサの戻り値は要素数 3 の float 型配列となっている。以下はその配列の中身を具体的に示した物である。

表 3　地磁気センサの戻り値

TYPE_MAGNETIC_FIELD	SensorEvents.value[0]	X 軸方向の地磁気強度
	SensorEvents.value[1]	Y 軸方向の地磁気強度
	SensorEvents.value[2]	Z 軸方向の地磁気強度

3.2　各ハードウェアが備えるソフトウェアにおける処理の流れ

このプログラムはまず、地磁気センサから情報を取得し、取得した値をあらかじめ準備しておいた配列へ代入、そして、配列の中身をグラフへ出力といった工程を無限ループで回すことによって常に最新の値がグラフへ出力されるようになっている。
以下はフローチャートである。

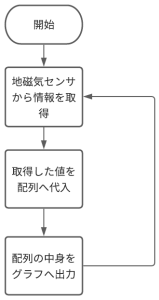


図 2　フローチャート

4　実行例

実行例 1 は適当な値をそれぞれの軸に設定した時の結果である。その後、それぞれのパラメータを自由に書き換えた後の結果が実行例 2 である。1、 2 より入力された値が正しく出力されることがわかる。

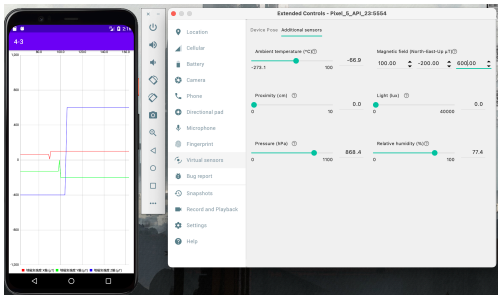


図 3　実行例 1

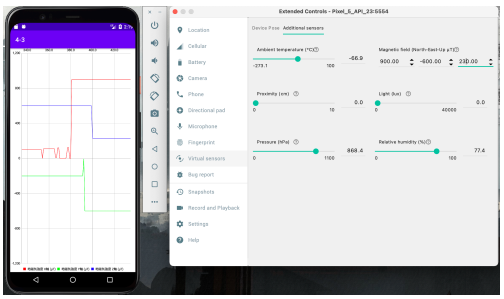


図 4　実行例 2

5　ソースコード

プログラムの説明はコード中にコメントアウトすることでおこなっている。
以下はソースコードである。

Listing 1　MainActivity.jave

```
1  package com.example.a4_3;
2
```

```
3  import androidx.appcompat.app.AppCompatActivity;
4
5  import android.content.Context;
6  import android.hardware.Sensor;
7  import android.hardware.SensorEvent;
8  import android.hardware.SensorEventListener;
9  import android.hardware.SensorManager;
10 import android.view.WindowManager;
11 import android.widget.TextView;
12
13 import java.util.List;
14
15 import android.os.Bundle;
16
17 import android.graphics.Color;
18 import android.util.Log;
19 import android.view.WindowManager;
20
21 import com.github.mikephil.charting.charts.LineChart;
22 import com.github.mikephil.charting.components.AxisBase;
23 import com.github.mikephil.charting.components.XAxis;
24 import com.github.mikephil.charting.components.YAxis;
25 import com.github.mikephil.charting.data.Entry;
26 import com.github.mikephil.charting.data.LineData;
27 import com.github.mikephil.charting.data.LineDataSet;
28 import com.github.mikephil.charting.formatter.IAxisValueFormatter;
29 import com.github.mikephil.charting.utils.Transformer;
30 import com.github.mikephil.charting.utils.ViewPortHandler;
31
32 public class MainActivity extends AppCompatActivity implements SensorEventListener {
33
34     private LineChart mLineChart;
35
36     // センサーマネージャを定義
37     private SensorManager manager;
38
39     // センサーから届いた値を格納する配列を定義
40     private float[] values = new float[3];
41
42     // グラフに表示するデータに関する値を定義
43     private int num; // グラフにプロットするデータの数
44
45     private String[] labels; // データのラベルを格納する配列
46     private int[] colors; // グラフにプロットする点の色を格納する配列
47     private float max, min; // グラフのY軸の最大値と最小値
48
49     // 値をプロットするx座標
```

```
50      private float count = 0;
51
52      @Override
53      protected void onResume() {
54          super.onResume();
55
56      // 情報を取得するセンサーの設定(今回は地磁気強度)
57          List<Sensor> sensors = manager.getSensorList(Sensor.TYPE_MAGNETIC_FIELD);
58          Sensor sensor = sensors.get(0);
59
60      // センサーからの情報の取得を開始
61          manager.registerListener(this, sensor, SensorManager.SENSOR_DELAY_UI);
62      }
63
64      // アプリケーション一時停止時に呼ばれるコールバック関数
65      @Override
66      protected void onPause() {
67          super.onPause();
68
69      // センサのリスナー登録解除
70          manager.unregisterListener(this);
71      }
72
73      // センサーイベント受信時に呼ばれるコールバック関数
74      public void onSensorChanged(SensorEvent event) {
75          // 取得した情報が地磁気強度センサーからのものか確認
76          if (event.sensor.getType() == Sensor.TYPE_MAGNETIC_FIELD) {
77          // 受け取った情報を格納用の配列にコピー
78              values = event.values.clone();
79          }
80      }
81
82      // センサーの精度の変更時に呼ばれるコールバック関数(今回は何もしない)
83      public void onAccuracyChanged(Sensor sensor, int accuracy) {
84      }
85
86      @Override
87      protected void onCreate(Bundle savedInstanceState) {
88          super.onCreate(savedInstanceState);
89          setContentView(R.layout.activity_main);
90
91          manager = (SensorManager) getSystemService(Context.SENSOR_SERVICE);
92
93          // アプリ実行中はスリープしない
94          getWindow().addFlags(WindowManager.LayoutParams.FLAG_KEEP_SCREEN_ON);
95
96          // グラフに表示するデータに関連する値を初期化
```

```
97          num = 3;
98          values = new float[num];
99          labels = new String[num];
100         colors = new int[num];
101
102         labels[0] = "地磁気強度 X軸 (μT)";
103         labels[1] = "地磁気強度 Y軸 (μT)";
104         labels[2] = "地磁気強度 Z軸 (μT)";
105
106         colors[0] = Color.rgb(0xFF, 0x00, 0x00); // 赤
107         colors[1] = Color.rgb(0x00, 0xFF, 0x00); // 緑
108         colors[2] = Color.rgb(0x00, 0x00, 0xFF); // 青
109
110         max = 1200;
111         min = -1200;
112
113         // グラフViewを初期化する
114         initChart();
115
116         // 一定間隔でグラフをアップデートする
117         new Thread(new Runnable() {
118             @Override
119             public void run() {
120                 while (true) {
121                     updateGraph();
122                     try {
123                         Thread.sleep(500);
124                     } catch (Exception e) {
125                         Log.e("Test", "例外出力", e);
126                     }
127                 }
128             }
129         }).start();
130     }
131
132     /**
133      * グラフViewの初期化
134      **/
135
136     private void initChart() {
137         // 線グラフView
138         mLineChart = (LineChart) findViewById(R.id.chart_DynamicMultiLineGraph);
139
140         // グラフ説明テキストを表示するか
141         mLineChart.getDescription().setEnabled(true);
142         // グラフ説明テキストのテキスト設定
143         mLineChart.getDescription().setText("Line Chart of Sensor Data");
```

```
144         // グラフ説明テキストの文字色設定
145         mLineChart.getDescription().setTextColor(Color.BLACK);
146         // グラフ説明テキストの文字サイズ設定
147         mLineChart.getDescription().setTextSize(10f);
148         // グラフ説明テキストの表示位置設定
149         mLineChart.getDescription().setPosition(0, 0);
150
151         // グラフへのタッチジェスチャーを有効にするか
152         mLineChart.setTouchEnabled(true);
153
154         // グラフのスケーリングを有効にするか
155         mLineChart.setScaleEnabled(true);
156
157         // グラフのドラッギングを有効にするか
158         mLineChart.setDragEnabled(true);
159
160         // グラフのピンチ/ズームを有効にするか
161         mLineChart.setPinchZoom(true);
162
163         // グラフの背景色設定
164         mLineChart.setBackgroundColor(Color.WHITE);
165
166         // 空のデータをセットする
167         mLineChart.setData(new LineData());
168
169         // Y軸(左)の設定
170         // Y軸(左)の取得
171         YAxis leftYAxis = mLineChart.getAxisLeft();
172         // Y軸(左)の最大値設定
173         leftYAxis.setAxisMaximum(max);
174         // Y軸(左)の最小値設定
175         leftYAxis.setAxisMinimum(min);
176
177         // Y軸(右)の設定
178         // Y軸(右)は表示しない
179         mLineChart.getAxisRight().setEnabled(false);
180
181         // X軸の設定
182         XAxis xAxis = mLineChart.getXAxis();
183         // X軸の値表示設定
184         xAxis.setValueFormatter(new IAxisValueFormatter() {
185             @Override
186             public String getFormattedValue(float value, AxisBase axis) {
187                 if (value >= 10) {
188                     // データ20個ごとに目盛りに文字を表示
189                     if (((int) value % 20) == 0) {
190                         return Float.toString(value);
```

```
                    }
                }
                // nullを返すと落ちるので、値を書かない場合は空文字を返す
                return "";
            }
        });
    }

    private void updateGraph() {
        // 線の情報を取得
        LineData lineData = mLineChart.getData();
        if (lineData == null) {
            return;
        }

        LineDataSet[] lineDataSet = new LineDataSet[num];

        for (int i = 0; i < num; i++) {
            // i番目の線を取得
            lineDataSet[i] = (LineDataSet) lineData.getDataSetByIndex(i);
            // i番目の線が初期化されていない場合は初期化する
            if (lineDataSet[i] == null) {
                // LineDataSetオブジェクト生成
                lineDataSet[i] = new LineDataSet(null, labels[i]);
                // 線の色設定
                lineDataSet[i].setColor(colors[i]);
                // 線にプロット値の点を描画しない
                lineDataSet[i].setDrawCircles(false);
                // 線にプロット値の値テキストを描画しない
                lineDataSet[i].setDrawValues(false);
                // 線を追加
                lineData.addDataSet(lineDataSet[i]);
            }
            // i番目の線に値を追加
            lineData.addEntry(new Entry(count, values[i]), i);
        }

        // 値更新通知
        mLineChart.notifyDataSetChanged();

        // X軸に表示する最大のEntryの数を指定
        mLineChart.setVisibleXRangeMaximum(100);

        // オシロスコープのように古いデータを左に寄せていくように表示位置をずらす
        mLineChart.moveViewTo(count, getVisibleYCenterValue(mLineChart), YAxis.
            AxisDependency.LEFT);

```

```
237             count++;
238         }
239
240         private float getVisibleYCenterValue(LineChart lineChart) {
241             Transformer transformer = lineChart.getTransformer(YAxis.AxisDependency.LEFT);
242             ViewPortHandler viewPortHandler = lineChart.getViewPortHandler();
243
244             float highestVisibleY = (float) transformer.getValuesByTouchPoint(
                    viewPortHandler.contentLeft(),
245                     viewPortHandler.contentTop()).y;
246             float highestY = Math.min(lineChart.getAxisLeft().mAxisMaximum,
                    highestVisibleY);
247
248             float lowestVisibleY = (float) transformer.getValuesByTouchPoint(
                    viewPortHandler.contentLeft(),
249                     viewPortHandler.contentBottom()).y;
250             float lowestY = Math.max(lineChart.getAxisLeft().mAxisMinimum, lowestVisibleY)
                    ;
251
252             return highestY - (Math.abs(highestY - lowestY) / 2);
253         }
254     }
```